Load-sensitive CPU Power Management for Web Search Engines

Matteo Catena[1,3], Craig Macdonald[2], Nicola Tonellotto[3]
[1] Gran Sasso Science Institute, 67100 L'Aquila, Italy
[2] University of Glasgow, Glasgow, G12 8QQ, UK
[3] National Research Council of Italy, 56124 Pisa, Italy
{matteo.catena, nicola.tonellotto}@isti.cnr.it, craig.macdonald@glasgow.ac.uk

ABSTRACT

Web search engine companies require power-hungry data centers with thousands of servers to efficiently perform searches on a large scale. This permits the search engines to serve high arrival rates of user queries with low latency, but poses economical and environmental concerns due to the power consumption of the servers. Existing power saving techniques sacrifice the raw performance of a server for reduced power absorption, by scaling the frequency of the server's CPU according to its utilization. For instance, current Linux kernels include frequency governors i.e., mechanisms designed to dynamically throttle the CPU operational frequency. However, such general-domain techniques work at the operating system level and have no knowledge about the querying operations of the server. In this work, we propose to delegate CPU power management to search engine-specific governors. These can leverage knowledge coming from the querying operations, such as the query server utilization and load. By exploiting such additional knowledge, we can appropriately throttle the CPU frequency thereby reducing the query server power consumption. Experiments are conducted upon the TREC ClueWeb09 corpus and the query stream from the MSN 2006 query log. Results show that we can reduce up to ~24% a server power consumption, with only limited drawbacks in effectiveness w.r.t. a system running at maximum CPU frequency to promote query processing quality.

Categories and Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval
Keywords: Power Consumption, CPU Frequency Scaling, Search Engines

1. INTRODUCTION

Nowadays search engines are a fundamental part of the Web, due to its enormous size. In fact, most users turn to a search engine to look for the information they need, producing billions of searches every day. And yet, users want to quickly receive answers to their questions, and are not willing to wait long for queries to be served [15]. To achieve low latencies, user queries are processed in a distributed fashion by thousands of query servers, which are organized in clusters and hosted by a data center [1]. This distributed architecture promotes raw performance but also raises environmental and economical issues. In fact, data centers have high electricity consumption due to servers and telecommunications. Moreover, as servers generate heat, data centers need additional energy for thermal cooling.

Research in Information Retrieval on energy-efficiency for search engines has only started comparatively recently. Chowdhury was the first to explicitly write about Green Information Retrieval and to propose a research agenda to reduce power consumption in IR systems [5]. Recently, the research community has proposed some approaches for improving the energy efficiency of search engines data centers. These can involve workload consolidation among multiple query servers [6], caching mechanisms [8, 12, 14] or query routing between different data centers [10, 18]. These works focus on the behavior of either the whole distributed infrastructure or the single data-centre of a Web search engine. Conversely, our work focuses on the energy efficiency optimization of a query server within the data-centre's cluster.

The power consumption of a typical server is dominated by its CPU. This is particularly true at low utilization levels, where CPUs consume a fixed amount of power without performing any demanding computations. For this reason, Hoelzle and Barroso [9] report the data centers' need for *energy-proportional computing*, i.e., for hardware components with power consumption proportional to utilization.

Dynamic Frequency Scaling (DFS) technologies sacrifices CPU performance for lower power consumptions, by throttling processor's frequency [17]. By doing so, CPUs operating at low frequencies absorb less power but have lower performance than CPUs working at higher frequencies. Operating system (OS) kernels can exploit DFS to achieve energy savings, for instance by throttling the server CPU frequency accordingly to the processor utilization [3]. When the processor is under-utilized, a low CPU frequency is selected. Conversely, a high CPU frequency is picked when the processor is heavily utilized by the system. However, the OS misses domain-specific information on the search engine software and its interactions with the incoming user queries. We advocate that this information can help improving the energy proportionality of a search engine, as identified in [9].

We propose search engine-specific frequency governors that can better adapt to varying query workloads by leveraging domain-specific information. While DFS states are typically

managed by the OS, their functionality can be (partially) controlled by application-level code [3]. We build upon this functionality to develop our proposed solution i.e., search engine-specific frequency governors which control the CPU frequency from within the query server. Indeed, knowledge of query server utilization and load facilitate a more refined control of the processor to achieve power savings.

Recently, Lo et al. propose a centralized feedback-based DFS controller for search clusters [11]. Their approach achieves considerable power savings by trading off performance, so that latency constraints are barely met for any workload. However, the authors report several challenges in deploying their centralized solution on large clusters. On the other hand, our approach is decentralized as it works at the single query server level.

The contribution of this paper are as follows: (1) we propose to exploit query servers' knowledge (e.g., utilization, load) to throttle the CPU frequency via search engine-specific frequency governors (2) we experimentally demonstrate that our solutions can achieve significant power savings without markedly damaging the query processing quality w.r.t. standard frequency governors.

2. PROBLEM STATEMENT

We model a query server as a first-come first-served queue, where incoming queries wait to be processed upon arrival to the search engine. As soon as a processing thread is available, it picks the next query from the queue and starts processing it, in disjunctive mode. Queries arrive to the system with an arrival rate λ and are processed at a processing rate μ (both expressed in qps, i.e., queries per second). The query arrival rate can vary over time, due to fluctuations in the query load [16]. The query processing rate may change as well, because of the DFS mechanism: for lower CPU frequencies we expect lower power consumption but also lower μ values, as the CPU speed is reduced and query processing takes longer.

Search engines users are impatient [15], so we assume that query servers must process queries within a short time threshold τ since their arrival, e.g., 1 second. However, since a query can spend some time waiting in the queue, processing threads may actually have less than τ seconds to solve certain queries. Additionally, execution times are variable and some queries may require more time than others to be processed [4]. If our system cannot complete a query processing within the time budget, the retrieval phase is terminated early and results computed so far are returned [10]. This partial processing will likely have a negative impact on the effectiveness of the returned results; however, they can benefit from subsequent effectiveness-improving processing stages, e.g., machine-learned ranking [19]. Conversely, when a query exceeds τ seconds waiting in the queue, the server just drops it. In this case, the system returns no results.

The power consumed by a query server (measured in Watts) can be divided into two components: a static part which is continuously consumed to operate the hosting machine, and a dynamic (or operational) part which depends on the CPU usage to perform query processing activities. In this work, we propose to exploit DFS technology to dynamically change the query server CPU frequency to reduce the operational power consumption. Indeed, as shown in Figure 1, the average operational power consumed by the query server is directly correlated to the CPU frequency. Of course, the operational power consumed by the server varies for different incoming workloads, since the machine load varies with the number of queries requiring processing. Clearly, lower CPU frequencies consume less power but also increases the number of unanswered and partially processed queries, as lower frequencies decrease the query processing rate μ.

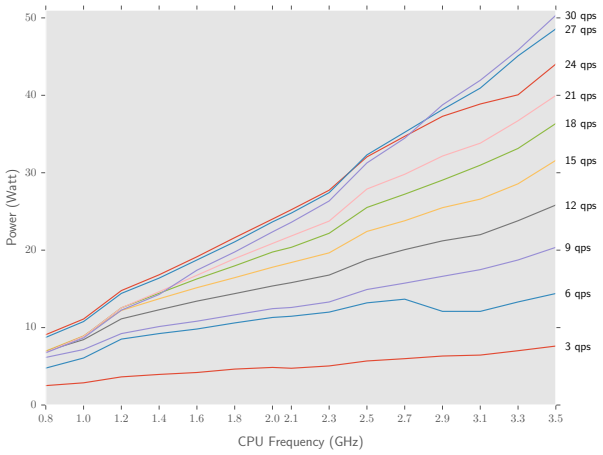

Figure 1: Average operational power (measured in Watts) consumed by a query server at different CPU frequencies and different workloads.

Hence, our goal is to reduce the power consumed by a query server to process queries, while providing an acceptable query processing quality.

3. PROPOSED SOLUTION

To achieve our goal, we propose to move CPU power management from the OS directly to the query server application code. The Linux OS leverages DFS to reduce power consumption. A CPU exposes a finite set of available operational frequencies to the OS kernel, which exploits software modules called *frequency governors* [3] to dynamically select the current operational frequency. OS governors vary the processor frequency according to metrics like the CPU utilization, i.e., the fraction of time a processor is busy performing computations. For instance, the `os-conservative` governor steps up the processor frequency when the CPU utilization is above a tunable threshold α (e.g., 0.8). Conversely, the CPU frequency is scaled down if utilization is below a tunable threshold β (e.g., 0.2). In any case, the OS governors select the CPU frequency using kernel-level information. They do not use application-specific information from the search engine, which we argue can help to improve the system energy efficiency. For this reason, we propose to delegate frequency scaling decisions to application-level modules, by implementing governors inside the search engine.

We develop two search engine-specific frequency governors, based on query server knowledge: `se-conservative` and `se-load`. The `se-conservative` governor is inspired by the `os-conservative` module, but it exploits the query server utilization instead of the raw CPU utilization. By using the query arrival and processing rates, the query server utilization ρ is computed as

$$\rho = \frac{\lambda}{k \cdot \mu} \tag{1}$$

where k is the number of threads processing queries [7]. The idea behind this governor is to maintain an acceptable query server utilization (e.g., 0.7 [7]) so that incoming queries can be easily processed without consuming too much power. Periodically, `se-conservative` computes the query server utilization and adjusts the CPU frequency. Frequency throttling is performed if ρ is above (resp. below) the tunable threshold α (resp. β). If an adjustment is required, `se-conservative` changes the processor frequency to obtain the desired utilization. The governor assumes that it will receive in the immediate future the same number of queries received during the last period. Using Equation (1), it computes the query processing rate necessary to obtain the target utilization. Finally, this governor selects the lowest frequency capable of producing such query processing rate, assuming processing rate directly proportional to CPU speed.

Our second governor, `se-load`, bases its frequency scaling decisions upon the number of queries N populating the query server, i.e., the queries currently queued or being processed. Given N, we define the query server load ℓ as:

$$\ell = \frac{N}{k} \tag{2}$$

Here, the principle is to reduce the query population in the server as fast as possible, when the query server load is too high (e.g. $\ell > 0.7$). Periodically, `se-load` observes the query server load and accordingly adjust the CPU frequency. If ℓ is greater than α, the processor is set to its maximum frequency. When ℓ is below β, the CPU frequency is stepped down from its current frequency to the next smaller one.

In the following section, we experiment in order to evaluate (a) how much power can be saved by using the search engine-specific frequency governors and (b) the corresponding impact on query processing quality.

4. EXPERIMENTAL SETUP

Experiments are conducted using the Terrier IR platform [13]. The platform is hosted on a dedicated Ubuntu 14.04 server; Linux kernel version is 3.13.0-45-generic. The machine is equipped with 32GB RAM and an 8-core Intel i7-4770K processor, which exposes 16 operational frequencies ranging from 800 MHz to 3.5 GHz. The ClueWeb09 (Cat. B) document collection is indexed to represent the first tier of a Web search engine. Stopwords are removed and the Porter stemmer is applied to all terms. The index stores document identifiers and term frequencies. The index is compressed with Elias-Fano encoding [20], and is kept in memory, shared among 8 query processing threads.

Queries are taken from the MSN 2006 query log and are submitted in real time to our system, while halving their original interarrival time. Since we use the first day of the dataset, every experiment take 12 hours to run; the average query load in 11.28 qps instead of the original 5.14, with a peak of 44 qps instead of 28.

For each query, we use BM25 to retrieve the top 1000 documents using WAND [2]. Upon arrival, queries are queued and have 1 second to be processed[1]. When this time expires before processing completion, query processing is early terminated and partial results are returned. If a query spends all its time in the queue, the system drops the query. The query is unanswered and an empty result list is returned.

[1]Approximatively 7% of the queries take at least one second to be solved.

For each experiment, we measure (a) the % of unanswered queries (%UQ), (b) the mean recall, relatively to an ideal system which has infinite time to process every query (RR) and (c) the power consumed by the query server (P). In particular, power consumption is measured at the server power socket by using an Alciom PowerSpy2 wattmeter. Measurements consider only the dynamic power consumption, i.e., we remove the power consumed by the server when idle ($\sim$41.8 Watt). Power is measured every 30 milliseconds and the mean value is reported for each experiment.

Our baselines are given by standard Linux frequency governors. In particular, we compare our approach to two baselines: `os-performance` which processes every query at the maximum CPU frequency; and `os-conservative` which adjusts the CPU frequency based on the processor utilization.

For `os-conservative`, frequency throttling decisions are taken every 0.08 milliseconds (the default value). Instead, our governors take decisions at every second, since we observe from the query log that the query arrival rate fluctuates every $\sim$1.2 seconds in average.

While `os-performance` is parameterless, other governors are tested under two different configurations. One has relaxed thresholds ($\alpha = 0.8$, $\beta = 0.2$ – as in the `os-conservative` default setting), so that the query server is likely to maintain a certain CPU frequency for longer periods. The other configuration has tighter thresholds (selected as $\alpha = 0.8$, $\beta = 0.6$), so that the query server will promptly react to changes in utilization or load.

We consider our search engine-specific governors successful if they show reduced power consumption (P) than the baselines, without marked degradation to query processing quality, as measured by relative recall (RR) and % of unanswered queries (%UQ).

5. EXPERIMENTAL ANALYSIS

Governor	α	β	%UQ	RR	P
os-performance	-	-	0.342	**0.931**	41.765
os-conservative	0.8	0.2	**0.283**	0.929	38.569
	0.8	0.6	0.295	0.927	35.381
se-conservative	0.8	0.2	0.352	0.911	36.016
	0.8	0.6	0.315	0.900	**31.727**
se-load	0.8	0.2	0.312	0.913	35.455
	0.8	0.6	0.292	0.912	32.888

Table 1: Percentage of unanswered queries (%UQ), mean relative recall (RR) and mean consumed power (P, in Watt) for different frequency governors under various settings of α, β.

Experiments results are reported in Table 1. We observe that the `os-performance` baseline consumes the highest operational power ($\sim$ 42 Watt) and causes the query server to drop more than 0.3% of the incoming queries. However, under this configuration the system shows the best relative recall (0.931). Relative recall values are statistically significant according to paired t-tests ($p < 0.01$).

The query server using the `os-conservative` governor shows reduced power consumption w.r.t. a server equipped with the `os-performance` governor. Indeed, `os-conservative` can consume from $\sim$8% to $\sim$15% less power than a governor which always maintains the CPU at the maximum frequency.

Also, `os-conservative` drops less queries but provides a slightly worse relative recall.

Our first search engine-specific governor, `se-conservative`, leads to reduced power consumption if compared to `os-conservative` runs. In fact, our governor saves more than 6% in power consumption when relaxed thresholds are set ($\alpha = 0.8$, $\beta = 0.2$); and more than 10% using tight thresholds ($\alpha = 0.8$, $\beta = 0.6$). These power savings come at the price of small degradation in query processing quality: the percentage of unanswered queries increases by ∼6% while the relative recall decreases by almost 2%, if we compare `se-conservative` to `os-conservative` with tight thresholds. Under the relaxed threshold, `se-conservative` drops ∼24% more queries than `os-conservative`, while its relative recall diminishes of ∼3% in comparison. When compared to `os-performance`, `se-conservative` can help saving from ∼14% to ∼24% in power consumption. Relative recall decreases by slightly more than 2% when using relaxed thresholds, and by almost 3% with tight ones. The percentage of unanswered queries increases of ∼3% w.r.t. `os-performance` when `se-conservative` uses relaxed thresholds. However, dropped queries decrease by ∼8% when tight thresholds are set.

Our second governor, namely `se-load`, obtains power savings similar to the `se-conservative` governor, but with a better query processing quality. Indeed, `se-load` saves more than 8% in power consumption when compared to `os-conservative` with relaxed thresholds. However, the relative recall detriment is less than 2% and the unanswered queries increment by just ∼10%. When the governors are configured with tight thresholds, `se-load` saves 7% in power consumption w.r.t. `os-conservative`. At the same time, relative recall is damaged for less than 2% and no additional queries are dropped. When compared to `os-performance`, `se-load` saves from ∼15% to ∼21% in power consumption. Relative recall is damaged by ∼2% under both threshold configurations. Instead, the percentage of unanswered queries benefits from our governor. Under relaxed thresholds, `se-load` drops ∼9% less queries than `os-performance` and ∼15% less queries remain unanswered by using tight thresholds.

Overall, experiments confirm that our approach is successful, as the search engine-specific governors show reduced power consumption than the two baselines. In particular, `se-conservative` provides the highest power saving. Relative recall (RR) and percentage of unanswered queries (%UQ) are not markedly damaged, especially when `se-load` is used.

6. CONCLUSIONS

In this work, we advocate that search engines infrastructures can save power at query server level, by leveraging knowledge on the server querying operations. We develop two search engine-specific frequency governors, `se-conservative` and `se-load`, which perform processor frequency throttling according to the query server utilization and load. By extensive experimentation, we evaluate the benefits and drawbacks of our approaches, compared to standard OS-level frequency governors. We find that `se-conservative` can help saving up to ∼24% power w.r.t. a system which operates at maximum CPU frequency to promote query processing quality. Indeed, `se-conservative` damages by just ∼3% both relative recall and percentage of unanswered queries. When compared to systems that use more energy efficient configurations, we find that our governors can still save at least 7% in power consumption. This gain costs only a limited detriment in relative recall (less than 2%) when `se-load` is used. Moreover, such power savings are important at data center-level too. Indeed, reduced CPU frequencies reduces heat output, and thus reduces thermal cooling expenditure. Greater power savings can be achieved by accepting more substantial degradation in query processing quality.

7. REFERENCES

[1] L. A. Barroso, J. Dean, and U. Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28, 2003.

[2] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *CIKM*, 2003, pages 426–434.

[3] D. Brodowski. CPU frequency and voltage scaling code in the Linux kernel: Linux CPUFreq. `https://www.kernel.org/doc/Documentation/cpu-freq/index.txt`, Visited: 2015-02-16.

[4] S. Büttcher, C. L. A. Clarke and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*, 2010, MIT Press.

[5] G. Chowdhury. An agenda for green information retrieval research. *Information Processing & Management*, 48(6):1067–1077, 2012.

[6] A. Freire, C. Macdonald, N. Tonellotto, I. Ounis, and F. Cacheda. A Self-adapting Latency/Power Tradeoff Model for Replicated Search Engines. In *WSDM*, 2014, pages 13–22.

[7] M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[8] N. Hidalgo, E. Rosas, V. Gil-Costa, and M. Marin. Assessing Energy Efficiency in ISP and Web Search Engine Collaboration. In *WAINA*, 2014, pages 299–304.

[9] U. Hoelzle and L. A. Barroso. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[10] E. Kayaaslan, B. B. Cambazoglu, R. Blanco, F. P. Junqueira, and C. Aykanat. Energy-price-driven Query Processing in Multi-center Web Search Engines. In *SIGIR*, 2011, pages 983–992.

[11] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso and C. Kozyrakis. Towards energy proportionality for large-scale latency-critical workloads. In *ISCA*, 2014, pages 301–312.

[12] M. Marin, V. Gil-Costa, and C. Gomez-Pantoja. New Caching Techniques for Web Search Engines. In *HPDC*, 2010, pages 215–226.

[13] C. Macdonald, R. McCreadie, R. Santos, and I. Ounis. From Puppy to Maturity: Experiences in Developing Terrier. In *OSIR Workshop*, 2012.

[14] F. B. Sazoglu, B. B. Cambazoglu, R. Ozcan, I. S. Altingovde, and O. Ulusoy. A Financial Cost Metric for Result Caching. In *SIGIR*, 2013, pages 873–876.

[15] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *Proc. Velocity*, 2009, page 1.

[16] F. Silvestri. *Mining Query Logs: Turning Search Usage Data into Knowledge*. Foundations and Trends in IR, Now Publishers Inc., 2010.

[17] D. C. Snowdon, S. Ruocco, and G. Heiser. Power Management and Dynamic Voltage Scaling: Myths and Facts. In *PARC Workshop*, 2005.

[18] A. Teymorian, O. Frieder, and M. A. Maloof. Rank-energy Selective Query Forwarding for Distributed Search Systems. In *CIKM*, 2013, pages 389–398.

[19] N. Tonellotto, C. Macdonald, and I. Ounis. Efficient and effective retrieval using selective pruning. In *WSDM*, 2013, pages 63–72.

[20] S. Vigna. Quasi-succinct indices. In *WSDM*, 2013, pages 83–92.